Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

SWAT - Sistema Web para Avaliação de Trabalhos

Aluno: Kayran dos Santos
Matricula: 08.1.4006

Orientador: David Menotti

Ouro Preto
15 de setembro de 2011

Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

SWAT - Sistema Web para Avaliação de Trabalhos

Proposta de monografia apresentada ao curso de Bacharelado em Ciência da Computação, Universidade Federal de Ouro Preto, como requisito parcial para a conclusão da disciplina Monografia II (BCC391).

Aluno: Kayran dos Santos
Matricula: 08.1.4006

Orientador: David Menotti

Ouro Preto
15 de setembro de 2011

Resumo

Apresentamos uma proposta de continuação do desenvolvimento do Sistema Web para Avaliação de Trabalhos, a ser aplicado inicialmente na disciplina de Estrutura de Dados I (EDI), com grande possibilidade de expansão para outras disciplinas do ciclo básico da Computação voltadas para a programação. A proposta apresentada auxiliará no desenvolvimento do novo plano pedagógico do DECOM no que diz respeito à otimização dos recursos didáticos.

Palavras-chave: Avaliação Automática, Trabalhos Práticos, Sistema Web.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Programas

1 Introdução

Uma das bases do aprendizado em Ciência da Computação é a confecção de Trabalhos Práticos (TPs) e resolução de listas de exercícios voltadas para programação. A disciplina de Estrutura de Dados I (EDI) utiliza esta base profundamente, exigindo dos alunos a entrega de no mínimo 3 TPs e inúmeros exercicios, sendo compostos de código fonte que resolva o problema proposto e documentação explicativa, aplicando os conceitos aprendidos na disciplina durante o período. A correção desses trabalhos é de extrema importância para o acompanhamento dos alunos em seu desenvolvimento na disciplina e gera um gasto de tempo enorme por parte do professor.

O grande número de alunos aumenta a quantidade de TPs e exercícios a serem corrigidos e dificulta ainda mais esta correção, além disso existem prazos a serem cumpridos estabelecidos pela universidade. Automatizar o processo de correção do código fonte com um sistema online auxiliaria os professores a cumprir os prazos estabelecidos e facilitaria o acompanhamento do andamento dos alunos por eles mesmos e pelo professor.

2 Justificativa

Grande parte das disciplinas do curso de Ciência da Computação se desenvolve a partir de atividades extra classe, os chamados Trabalhos Práticos. As correções destes TPs são de extrema importância tanto para o aluno saber como anda seu desenvolvimento na disciplina, como para o professor acompanhar o desenvolvimento da turma. Porém estas correções demandam considerável tempo do professor e ainda existe o risco de demorar mais do que é permitido pelas normas da universidade. O sistema Web proposto vem para auxiliar o professor e acelerar o processo de avaliação de TPs, e vem ao encontro do novo projeto pedagógico do curso de Ciência da Computação, no que diz respeito à otimização dos recursos didáticos.

3 Objetivos

3.1 Objetivo geral

O objetivo deste trabalho é desenvolver um sistema acessível pela internet para auxiliar a correção de trabalhos práticos e exercícios da disciplina de Estrutura de Dados I com possível expansão para outras disciplinas. O sistema será o responsável pela correção do código fonte do aluno que será submetido em uma página Web, verificando se o mesmo compila e executa corretamente, gerando uma saída correspondente com a saída esperada para as várias instâncias de teste que serão criadas para cada TP.

3.2 Objetivos específicos

Os objetivos específicos são:

- Configuração de um servidor para armazenamento dos dados dos alunos, dos trabalhos submetidos e das páginas desenvolvidas;
- Elaboração do Sistema Web para Avaliação de Trabalhos (SWAT) seguindo a metologia apresentada na seção seguinte;
- Execução de testes do sistema e monitoramento de seu funcionamento;
- Acompanhamento dos alunos durante a submissão de seus TPs.

4 Metodologia

A metodologia adotada para o desenvolvimento do sistema será:

1. Em um primeiro momento serão revisados a análise de requisitos do sistema, o modelo Entidade - Relacionamento para o banco de dados seguindo os conceitos apresentados em [10], o esquema desenvolvido para o banco e a esquematização do sistema utilizando MoLIC[12, 15];

2. Em seguida os esquemas serão seguidos de forma a implementar o sistema utilizando tecnologias Web[14, 13, 6], em um banco de dados PostgreSQL[11];

3. Após haverá um período de testes do sistema em meio à disciplina de EDI com a submissão supervisionada de um TP;

4. Depois do período de testes na disciplina de EDI, o mesmo será aplicado em larga escala na disciplina.

4.1 Tecnologias Utilizadas

O sistema será desenvolvido utilizando as tecnologias PHP (Personal Home Page)[5, 6] como linguagem *backend* para o Flex SDK[4, 2], CSS (Cascading Style Sheets)[3] para definir os estilos das páginas do sistema, tudo isso interligado com um banco de dados PostgreSQL[7] para armazenamento dos dados dos alunos. Este será hospedado em um servidor do DECOM configurado com Linux e servidor Apache.

4.2 Organização do Sistema

O sistema será desenvolvido utilizando algumas ideias do BOCA (BOCA Online Contest Administrator) [9], sistema de submissão e correção automática utilizada pela ACM em sua Maratona de Programação [1, 8].

Basicamente o sistema será composto de um sistema Web de submissão de arquivos a ser acessado a partir de um "login". O aluno efetuará "login" no sistema a partir de seu Registro Acadêmico (RA), também conhecido como número de matrícula. Após o "login" o aluno terá acesso, principalmente, a uma página de perfil simples, uma página de submissão dos arquivos e uma página com o histórico de suas submissões. Vale ressaltar que apenas o aluno, o professor e eventualmente um monitor da disciplina terão acesso às notas.

O arquivo submetido na página será armazenado no servidor até o fim do período para possíveis esclarecimentos. Após a submissão do arquivo compactado, este mesmo será descompactado e compilado usando regras pré-estabelecidas a serem decididas e apresentadas aos alunos. A compilação vai gerar um arquivo executável Linux, que vai executar sobre uma base de testes preparada para cada problema. Esta base de testes será composta por arquivos com possíveis entradas ao programa, que estarão padronizadas, sendo este padrão apresentado aos alunos assim que for apresentado um TP. O programa vai gerar uma saída de acordo com a entrada utilizada. Esta saída vai ser comparada com a saída esperada usando comandos do *shell* Linux. Caso a saída seja a esperada, será atribuída ao TP do aluno nota total no que tange à execução do programa. Caso o programa apresente algum problema na compilação exista algum

problema de execução (Estouro de tempo, Saída mal formatada, Saída com valores errados), este será retornado ao aluno pela página Web após a supervisão do resultado por uma pessoa designada para tal. Para caso do estouro de tempo, ao ser definido o problema a ser resolvido será definido tempo limite para a execução do mesmo.

Será possível ao aluno fazer submissões que não terão nota associada, permitindo ao aluno uma certa quantidade a ser definida de submissões para o sistema, onde apenas receberá um *feedback* de erros em seu código.

Todas as submissões definitivas de TPs serão avaliadas por um supervisor delegado pelo professor da disciplina caso o sistema acuse um erro automático. Isto será feito para que mais de um erro seja detectado em paralelo e um *feedback* mais completo seja repassado ao aluno. Submissões não definitivas de TP e submissões de exercícios (Listas) não serão supervisionadas.

O perfil do aluno será composto apenas de informações básicas para identificação do mesmo, sendo que estas informações também serão disponíveis apenas para o professor e poderão ser atualizadas apenas pelo aluno. Além disso ele terá a página de histórico, onde os resultados dos TPs anteriores estarão disponíveis para que o mesmo possa controlar seu desempenho nos TPs.

O sistema será organizado de forma que possam ser criadas disciplinas, cada disciplina tutorada por um professor previamente cadastrado no sistema. A cada semestre poderá ser criado um perfil de disciplina para que novas atividades possam ser criadas. Os alunos cadastrados no sistema serão vinculados a uma ou mais disciplinas cadastradas.

Para cada disciplina poderá existir um monitor,sendo que este será responsável por supervisionar as submissões de tarefas de alunos da disciplina à qual estão vinculados. As tarefas serão vinculadas a uma disciplina, portanto apenas alunos que cursam a mesma terão acesso a seus dados, podendo submeter soluções a elas, da mesma forma apenas os professores das disciplinas poderão criar tarefas para a disciplina e apenas os monitores destas disciplinas poderão supervisionar submissões de tarefas da mesma.

A inserção de alunos no banco de dados poderá ser feita de modo semiautomático, por meio de um arquivo no formato CSV.

O sistema possuirá um usuário com privilégios diferenciados, chamado administrador (Admin). Este usuário será responsável pelo cadastro de novos professores que queiram utilizar o sistema, bem como pela criação de novas disciplinas no sistema. Será possível a este usuário também cadastrar alunos no sistema e efetuar a inscrição destes nas disciplinas já existentes. Ao administrador caberá também a remoção de dados, usuários e disciplinas do sistema quando necessário.

4.3 Período de Testes

Os testes do sistema inicialmente serão feitos com problemas simples e ainda no servidor do Laboratório de Processamento Digital de Imagens (LaPDI), durante o período de desenvolvimento. Após passar por estes testes preliminares, o mesmo será instalado no servidor do DECOM e disponibilizado para os alunos da disciplina de EDI exclusivamente pela internet. Ao disponibilizar o sistema, após entrar em acordo com o professor da disciplina, adaptaremos pelo menos um trabalho para testes do sistema. Depois de aprovado nestes testes, o sistema poderá ser incorporado à disciplina de EDI e poderemos estudar a sua implantação em outras disciplinas de programação básica.

5 Cronograma de atividades

As atividades a serem desenvolvidas são mostradas na lista abaixo:

1. Revisão da análise de requisitos e dos diagramas do sistema;
2. Desenvolvimento do sistema;
3. Testes preliminares;
4. Testar o sistema em um trabalho;
5. Resolução de problemas observados;
6. Aplicar o sistema em EDI;
7. Elaboração da Monografia;
8. Apresentação oral de monografia para banca;
9. Escrita de um artigo.

Na Tabela 1, apresenta-se o cronograma para desenvolvimento das atividades da lista. Cada coluna representa uma quinzena do mês.

Atividades	**Set/2**	**Out/1**	**Out/2**	**Nov/1**	**Nov/2**	**Dez/1**	**Dez/2**
1	X	X					
2	X	X	X				
3	X	X	X				
4		X	X	X			
5	X	X	X	X	X	X	X
6				X	X	X	X
7	X	X	X	X	X		
8					X	X	X
9				X	X	X	X

Tabela 1: Cronograma de Atividades.

5.1 Atividades Concluídas/Avançadas

As atividades do cronograma apresentado anteriormente que já possuem alguns resultados são a Revisão da análise de requisitos e dos diagramas do sistema e o desenvolvimento. Toda a modelagem de banco de dados está pronta, com os scripts para criação do banco em SQL construidos. Com relação à modelagem falta terminar a modelagem das interações, para que seja possível desenvolver as interfaces do sistema.

A primeira parte do sistema que está sendo desenvolvida é a persistência em banco. Foram feitos o Diagrama Entidade-Relacionamento (ER) (Figura 1) e o Modelo Relacional (Figura 2) para o banco de dados do sistema. A partir deste diagramas foi projeto o script de criação do Banco de Dados usando SQL (Programa 1) e alguns dos métodos de acesso ao banco de dados utilizando PHP (Programa 2, 3, 4, 5, 6, 7)

Referências

[1] The ACM-ICPC International Collegiate Programing Contest Web Site sponsored by IBM. `http://icpc.baylor.edu/`, 2011. Visitado em 15 de setembro de 2011.

[2] Adobe Flex - Wikipédia. `http://pt.wikipedia.org/wiki/Adobe_Flex`, 2011. Visitado em 15 de setembro de 2011.

[3] Cascading Style Sheets - Wikipédia. `http://pt.wikipedia.org/wiki/Cascading_Style_Sheets`, 2011. Visitado em 15 de setembro de 2011.

[4] open source framework, web application software development | Flex - Adobe. `http://www.adobe.com/products/flex/`, 2011. Visitado em 15 de setembro de 2011.

[5] PHP - Wikipédia. `http://pt.wikipedia.org/wiki/PHP`, 2011. Visitado em 15 de setembro de 2011.

[6] PHP: Hypertext Preprocessor. `http://www.php.net/`, 2011. Visitado em 15 de setembro de 2011.

[7] PostgreSQL: The world's most advanced open source database. `http://www.postgresql.org/`, 2011. Visitado em 15 de setembro de 2011.

[8] XVI Maratona de Programação. `http://maratona.ime.usp.br/`, 2011. Visitado em 15 de setembro de 2011.

[9] Cassio P. de Campos and Carlos E. Ferreira. Boca: um sistema de apoio a competicoes de programacao. *SBC Workshop de Educação em Computação*, aug 2004.

[10] Ramez Elmasri and Shamkant B. Navathe. *Sistema de Banco de Dados.* Addison Wesley, 4th edition, 2005.

[11] André Milani. *PostgreSQL Guia do Programador.* Editora Novatec, 2008.

[12] RO Prates and SDJ Barbosa. Introdução à teoria e prática da interação humano computador fundamentada na engenharia semiótica. In *JAI'2007 Jornada de Atualização em Informática*, 2007. Capítulo 8.

[13] Daniel Pace Schmitz. *Desenvolvendo Sistemas com Flex e PHP*. Editora Novatec, 2009.

[14] Maurício Samy Silva. *Construindo Sites com CSS e (X)HTML: sites controlados por folhas de estilo em cascata*. Editora Novatec, 2008.

[15] Élton José da Silva. *Sistemas Interativos*. Élton José da Silva, 2006. Capítulo 7, páginas 92-112.

Apêndice A - Diagramas

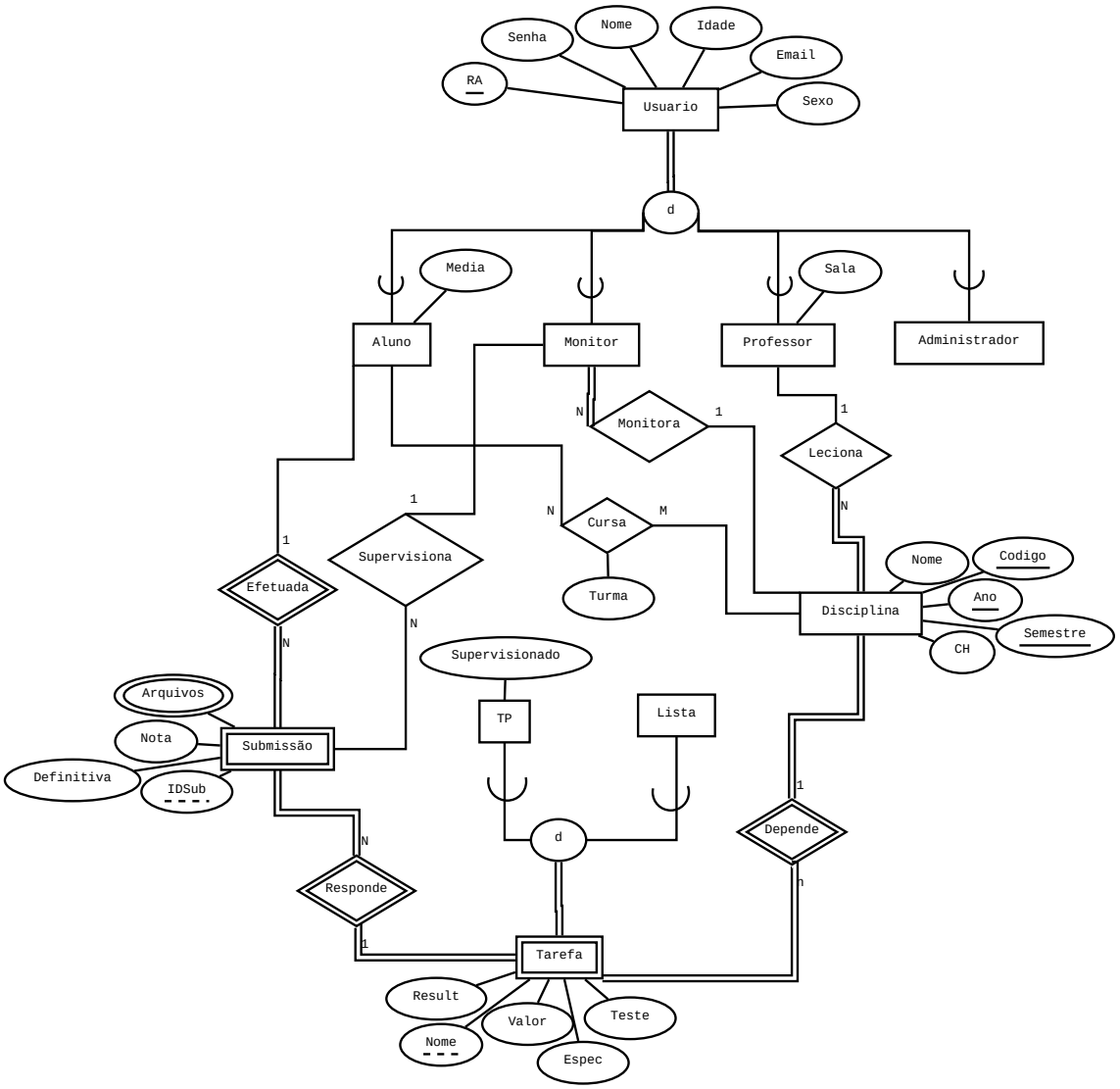

Figura 1: Diagrama ER

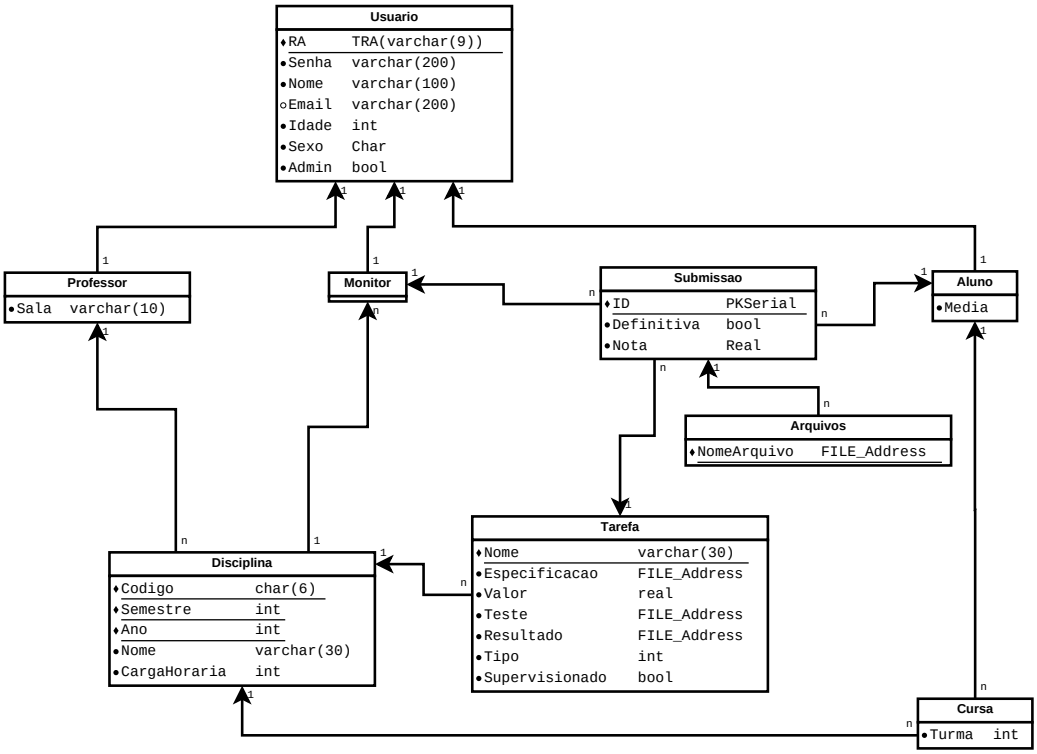

Figura 2: Modelo Relacional

Apendice B - Códigos

```
CREATE DATABASE SWAT;
BEGIN;

CREATE DOMAIN TRA as char(9);
CREATE DOMAIN TFILE as varchar(200);

CREATE TABLE USUARIO(
RA TRA UNIQUE NOT NULL,
SENHA VARCHAR(200) NOT NULL,
NOME VARCHAR(100) NOT NULL,
EMAIL VARCHAR(200),
IDADE INTEGER NOT NULL,
SEXO CHAR(1) NOT NULL,
ADMIN BOOLEAN NOT NULL DEFAULT FALSE,
CONSTRAINT USUPK PRIMARY KEY(RA),
CONSTRAINT SEXCHK CHECK(Sexo='M' OR sexo = 'F')
);

CREATE TABLE PROFESSOR(
RA TRA UNIQUE NOT NULL,
SALA VARCHAR(5),
CONSTRAINT PROFPK PRIMARY KEY(RA),
CONSTRAINT PROFUSUFK FOREIGN KEY(RA) REFERENCES USUARIO(RA) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE MONITOR(
RA TRA UNIQUE NOT NULL,
CONSTRAINT MONPK PRIMARY KEY(RA),
CONSTRAINT MONUSUFK FOREIGN KEY(RA) REFERENCES USUARIO(RA) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE ALUNO(
RA TRA UNIQUE NOT NULL,
MEDIA FLOAT,
CONSTRAINT ALUPK PRIMARY KEY(RA),
CONSTRAINT ALUUSUFK FOREIGN KEY(RA) REFERENCES USUARIO(RA) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE DISCIPLINA(
CODIGO CHAR(6) NOT NULL,
SEMESTRE INTEGER NOT NULL ,
ANO INTEGER NOT NULL,
NOME VARCHAR(30) NOT NULL,
CARGAHORARIA INTEGER NOT NULL,
RA_PROF TRA NOT NULL,
RA_MON TRA,
CONSTRAINT DISCPK PRIMARY KEY(CODIGO, SEMESTRE, ANO),
CONSTRAINT DISCPROFFK FOREIGN KEY(RA_PROF) REFERENCES PROFESSOR(RA) ON
    DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT DISCMONFK FOREIGN KEY(RA_MON) REFERENCES MONITOR(RA) ON DELETE
    CASCADE ON UPDATE CASCADE
```

```
);

CREATE TABLE CURSA(
CODIGO_DISC CHAR(6) NOT NULL,
SEMESTRE_DISC INTEGER NOT NULL ,
ANO_DISC INTEGER NOT NULL,
RA_ALU TRA NOT NULL,
TURMA INT NOT NULL,
CONSTRAINT CURPK PRIMARY KEY(CODIGO_DISC, SEMESTRE_DISC, ANO_DISC, RA_ALU
    ),
CONSTRAINT CURDISCFK FOREIGN KEY(CODIGO_DISC, SEMESTRE_DISC, ANO_DISC)
    REFERENCES DISCIPLINA(CODIGO, SEMESTRE, ANO) ON DELETE CASCADE ON
    UPDATE CASCADE,
CONSTRAINT CURALUFK FOREIGN KEY(RA_ALU) REFERENCES ALUNO(RA) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE TAREFA(
CODIGO_DISC CHAR(6) NOT NULL,
SEMESTRE_DISC INTEGER NOT NULL ,
ANO_DISC INTEGER NOT NULL,
NOME VARCHAR(30) NOT NULL,
ESPECIFICACAO TFILE NOT NULL,
VALOR FLOAT NOT NULL,
TESTE TFILE NOT NULL,
RESULTADO TFILE NOT NULL,
TIPO INT NOT NULL,
SUPERVISIONADO BOOLEAN NOT NULL DEFAULT FALSE,
CONSTRAINT TARPK PRIMARY KEY(CODIGO_DISC, SEMESTRE_DISC, ANO_DISC, NOME),
CONSTRAINT TARDISCFK FOREIGN KEY(CODIGO_DISC, SEMESTRE_DISC, ANO_DISC)
    REFERENCES DISCIPLINA(CODIGO, SEMESTRE, ANO) ON DELETE CASCADE ON
    UPDATE CASCADE,
CONSTRAINT TIPOCHK CHECK(TIPO = 1 OR TIPO = 2)
);

CREATE TABLE SUBMISSAO(
RA_ALU TRA NOT NULL,
CODIGO_DISC CHAR(6) NOT NULL,
SEMESTRE_DISC INTEGER NOT NULL ,
ANO_DISC INTEGER NOT NULL,
NOME_TAR VARCHAR(30) NOT NULL,
ID SERIAL UNIQUE,
DEFINITIVA BOOLEAN NOT NULL,
NOTA FLOAT NOT NULL,
RA_MON TRA,
CONSTRAINT SUBPK PRIMARY KEY(RA_ALU, CODIGO_DISC, SEMESTRE_DISC, ANO_DISC
    , NOME_TAR, ID),
CONSTRAINT SUBALUFK FOREIGN KEY(RA_ALU) REFERENCES ALUNO(RA) ON DELETE
    CASCADE ON UPDATE CASCADE,
CONSTRAINT SUBMONFK FOREIGN KEY(RA_MON) REFERENCES MONITOR(RA) ON DELETE
    CASCADE ON UPDATE CASCADE,
CONSTRAINT SUBTARFK FOREIGN KEY(CODIGO_DISC, SEMESTRE_DISC, ANO_DISC,
    NOME_TAR) REFERENCES TAREFA(CODIGO_DISC, SEMESTRE_DISC, ANO_DISC, NOME
    ) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE ARQUIVOS(
```

```
RA_ALU TRA NOT NULL,
CODIGO_DISC CHAR(6) NOT NULL,
SEMESTRE_DISC INTEGER NOT NULL ,
ANO_DISC INTEGER NOT NULL,
NOME_TAR VARCHAR(30) NOT NULL,
ID_SUB INTEGER NOT NULL,
NOME TFILE NOT NULL,
CONSTRAINT ARQPK PRIMARY KEY(RA_ALU, CODIGO_DISC, SEMESTRE_DISC, ANO_DISC
    , NOME_TAR, ID_SUB, NOME),
CONSTRAINT ARQSUBFK FOREIGN KEY(RA_ALU, CODIGO_DISC, SEMESTRE_DISC,
    ANO_DISC, NOME_TAR, ID_SUB) REFERENCES SUBMISSAO(RA_ALU, CODIGO_DISC,
    SEMESTRE_DISC, ANO_DISC, NOME_TAR, ID) ON DELETE CASCADE ON UPDATE
    CASCADE
);

END;
```

Programa 1: SQL de Criação do Banco

```
<?php

class SWATConecta {

    private static $host = "localhost";
    private static $port = "5432";
    private static $dbname = "swat";
    private static $usuario = "swat";
    private static $pass = "ufopdecomcom20lapdi";
    private static $con;
    private static $err;

    public static function conecta() {
        $str = 'host=' . self::$host . ' port=' . self::$port . ' dbname=
            ' . self::$dbname . ' user=' . self::$usuario . ' password=' .
             self::$pass;
        self::$con = pg_connect($str);
        pg_set_client_encoding(self::$con, "UNICODE");
    }

    public static function fecha() {
        pg_close(self::$con);
    }

    public static function iniciaTransacao() {
        SWATConecta::conecta();
        $query = "begin;";
        SWATConecta::executa($query);
        return SWATConecta::notice();
    }

    public static function desfazTransacao() {
        $query = "rollback;";
        SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        SWATConecta::fecha();
        return $msg;
    }
```

```
    public static function encerraTransacao() {
        $query = "end;";
        SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        SWATConecta::fecha();
        return $msg;
    }

    public static function executa($query) {
        $result = pg_query(self::$con, $query);
        self::$err = pg_errormessage(self::$con);
        return $result;
    }

    public static function notice() {
        return self::$err;
    }

    public static function horaSistema() {
        $hora = date('G:i:s');
        return $hora;
    }

    public static function dataSistema() {
        $data = date("d/m/Y");
        return $data;
    }

    public static function diaSistema() {
        $numdia = strftime('%u');
        switch ($numdia) {
            case 1:$dia = 'Segunda';
                break;
            case 2:$dia = 'Terca';
                break;
            case 3:$dia = 'Quarta';
                break;
            case 4:$dia = 'Quinta';
                break;
            case 5:$dia = 'Sexta';
                break;
            case 6:$dia = 'Sabado';
                break;
            case 7:$dia = 'Domingo';
                break;
        }
        return $dia;
    }

}

?>
```

Programa 2: Conexão com o Banco

```
<?php
```

```php
include_once "SWATConecta.php";

class Admin {

    public static function criaDisc($codigo, $semestre, $ano,$nome,$ch,
        $RA_Prof,$RA_Mon){
        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "Erro Conexao Admin::criaDisc ";
        }
        $query = "INSERT INTO DISCIPLINA(CODIGO,SEMESTRE,ANO,NOME,
            CARGAHORARIA,RA_PROF,RA_MON)
            VALUES ('$codigo',$semestre,$ano,'$nome',$ch,'$RA_Prof','
                $RA_Mon')";
        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::desfazTransacao();
            return "erro Insercao Admin::Criadisc";
        }
        $msg = SWATConecta::encerraTransacao();
        if (strcmp($msg, ""))
            return "erro Final Admin::Criadisc";

    }
    public static function criaUsu($ra, $senha, $nome,$idade,$sexo,$tipo)
        {
        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "Erro Conexao Admin::criaUsu ";
        }
        $query = "INSERT INTO USUARIO(RA,SENHA,NOME,IDADE,SEXO,ADMIN)
            VALUES ('$ra','$senha','$nome',$idade,'$sexo','FALSE')";
        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::desfazTransacao();
            return "erro Insercao Admin::CriaUsu";
        }
        if (strcmp($tipo,"prof")){
            $query = "INSERT INTO PROFESSOR(RA,SALA)
            VALUES ('$ra','')";
        }else if (strcmp($tipo,"alu")){
            $query = "INSERT INTO ALUNO(RA,MEDIA)
            VALUES ('$ra',0.0)";
        }else if (strcmp($tipo,"mon")){
            $query = "INSERT INTO MONITOR(RA,EMAIL)
            VALUES ('$ra','')";
        }else{
            SWATConecta::desfazTransacao();
            return "erro Tipo Admin::CriaUsu";
        }
        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
```

```
55          if (strcmp($msg, "")) {
                SWATConecta::desfazTransacao();
                return "erro Insercao Admin::CriaUsu";
            }
            $msg = SWATConecta::encerraTransacao();
60          if (strcmp($msg, ""))
                return "erro Final Admin::CriaUsu";

        }

65      public static function matriculaAluno($codigo,$semestre,$ano,
            $ra_aluno,$turma){
            $msg = SWATConecta::iniciaTransacao();
            if (strcmp($msg, "")) {
                SWATConecta::encerraTransacao();
                return "Erro Conexao Admin::Matricula ";
70          }
            $query = "INSERT INTO CURSA(CODIGO_DISC,SEMESTRE_DISC,ANO_DISC,
                RA_ALUNO,TURMA)
                VALUES ('$codigo',$semestre,$ano,'$ra_aluno',$turma)";
            $result = SWATConecta::executa($query);
            $msg = SWATConecta::notice();
75          if (strcmp($msg, "")) {
                SWATConecta::desfazTransacao();
                return "erro Insercao Admin::Matricula";
            }
            $msg = SWATConecta::encerraTransacao();
80          if (strcmp($msg, ""))
                return "erro Final Admin::Matricula";
        }

        public static function login($RA_Admin,$senha){
85
            $msg = SWATConecta::iniciaTransacao();
            if (strcmp($msg, "")) {
                SWATConecta::encerraTransacao();
                return "Erro Conexao Admin::login ";
90          }
            $query = "SELECT *
            FROM USUARIO
            WHERE RA = $RA_Admin and SENHA = $senha";

95          $result = SWATConecta::executa($query);
            $msg = SWATConecta::notice();
            if (strcmp($msg, "")) {
                SWATConecta::encerraTransacao();
                return "erro Login Admin::Login";
100         }
            $response;
            $cont = 0;
            while ($data = pg_fetch_array($result)) {
                foreach ($data as $i => $value) {
105                 $data[$i] = utf8_decode($value);
                }
                $response[] = $data;
                $cont++;
            }
```

```
110         $msg = SWATConecta::encerraTransacao();
            if (strcmp($msg, ""))
                return "erro Final Admin::Login";
            if ($cont > 0)
                return $response;
115         return "Nome de Usuario ou Senha Invalido";

        }

    }
120
    ?>
```

Programa 3: Classe Admin

```
<?php

include_once "SWATConecta.php";

5   class Aluno {

        public static function submete($RA_aluno, $codigo_disc,
            $semestre_disc, $ano_disc, $nome_tarefa, $Lista_filesAdress,
            $final) {
            $msg = SWATConecta::iniciaTransacao();
            if (strcmp($msg, "")) {
10              SWATConecta::encerraTransacao();
                return "Erro Conexao Aluno::submete ";
            }
            $query = "INSERT INTO SUBMISSAO (RA_ALU,CODIGO_DISC,SEMESTRE_DISC
                ,ANO_DISC,NOME_TAR,DEFINITIVA)
                        VALUES('$RA_aluno','$codigo_disc',$semestre_disc,
                            $ano_disc,'$nome_tarefa',$final);";
15
            $result = SWATConecta::executa($query);
            $msg = SWATConecta::notice();
            if (strcmp($msg, "")) {
                SWATConecta::desfazTransacao();
20              return "erro Insercao Aluno::submete";
            }

            $query = "SELECT MAX(ID) FROM SUBMISSAO";

25          $result = SWATConecta::executa($query);
            $msg = SWATConecta::notice();
            if (strcmp($msg, "")) {
                SWATConecta::desfazTransacao();
                return "erro GetID Aluno::submete";
30          }
            $response1;
            $cont = 0;
            while ($data = pg_fetch_array($result)) {
                $response1[] = $data;
35              $cont++;
            }
            if ($cont == 0) {
                SWATConecta::desfazTransacao();
```

```
            return "null 1";
        }
        $id = $response1[0][0];

        foreach ($Lista_filesAdress as $fileadress) {
            $query = "INSERT INTO ARQUIVOS (RA_ALU,CODIGO_DISC,
                SEMESTRE_DISC,ANO_DISC,NOME_TAR,NomeArquivo)
                    VALUES('$RA_aluno','$codigo_disc',$semestre_disc,
                        $ano_disc,'$nome_tarefa',$fileadress);";
            $result = SWATConecta::executa($query);
            $msg = SWATConecta::notice();
            if (strcmp($msg, "")) {
                SWATConecta::desfazTransacao();
                return "erro Insercao arquivos Aluno::submete";
            }
        }
        $msg = SWATConecta::encerraTransacao();
        if (strcmp($msg, ""))
            return "erro Final Aluno::Historico";
    }

    public static function historico($RA_Aluno){
        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "Erro Conexao Aluno::historico ";
        }
        $query = "SELECT CODIGO_DISC, SEMESTRE_DISC,ANO_DISC,NOME_TAR,
            NOTA, Definitiva
        FROM SUBMISSAO
        WHERE RA_Aluno = $RA_Aluno";

        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "erro Busca Aluno::Historico";
        }
        $response;
        $cont = 0;
        while ($data = pg_fetch_array($result)) {
            foreach ($data as $i => $value) {
                $data[$i] = utf8_decode($value);
            }
            $response[] = $data;
            $cont++;
        }
        $msg = SWATConecta::encerraTransacao();
        if (strcmp($msg, ""))
            return "erro Final Aluno::Historico";
        if ($cont > 0)
            return $response;
        return "Nao ha Submissoes!";
    }

    public static function login($RA_Aluno,$senha){
```

```
        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "Erro Conexao Aluno::historico ";
        }
        $query = "SELECT *
        FROM USUARIO
        WHERE RA = $RA_Aluno and SENHA = $senha";

        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "erro Login Aluno::Login";
        }
        $response;
        $cont = 0;
        while ($data = pg_fetch_array($result)) {
            foreach ($data as $i => $value) {
                $data[$i] = utf8_decode($value);
            }
            $response[] = $data;
            $cont++;
        }
        $msg = SWATConecta::encerraTransacao();
        if (strcmp($msg, ""))
            return "erro Final Aluno::Login";
        if ($cont > 0)
            return $response;
        return "Nome de Usuario ou Senha Invalido";

    }

}

?>
```

Programa 4: Classe Aluno

```
<?php

include_once "SWATConecta.php";

class Monitor {

    public static function buscaSubmissoes($ra_mon){
        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "erro Conexao Monit::buscaSubmissoes";
        }
        $query = "SELECT CODIGO,SEMESTRE,ANO FROM DISCIPLINA WHERE RA_MON
            = $ra_mon;";
        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
```

```
            return "erro BUSCADISC Monit::BuscaSubmissoes";
        }
        $data = pg_fetch_array($result);
        $codigo = utf8_decode($data[0]);
        $semestre = $data[1];
        $ano = $data[2];
        if($codigo==null || $semestre == null || $ano == null){
            SWATConecta::encerraTransacao();
            return "erro BuscaDisc Monit::BuscaSubmissoes";
        }
        $query = "SELECT NOME,ESPECIFICACAO,VALOR,TESTE,RESULTADO
        FROM TAREFA
        WHERE CODIGO_DISC = '$codigo' AND
        SEMESTRE_DISC = $semestre AND
        ANO_DISC = $ano AND
        SUPERVISIONADO = 'TRUE'";
        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "erro BuscaTarefa Monit::BuscaSubmissoes";
        }
        $cont = 0;
        while ($data = pg_fetch_array($result)) {
            foreach ($data as $i => $value) {
                $data[$i] = utf8_decode($value);
            }
            $response[] = $data;
            $cont++;
        }
        if(cont<=0){
            SWATConecta::desfazTransacao();
            return "erro BuscaTarefa Monit::BuscaSubmissoes";
        }
        foreach($response as $tarefa){
            $query = "SELECT ID";
        }
    }

    public static function login($RA_Monit,$senha){

        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "Erro Conexao Monit::historico ";
        }
        $query = "SELECT *
        FROM USUARIO
        WHERE RA = $RA_Monit and SENHA = $senha";

        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "erro Login Monit::Login";
        }
        $response;
```

```
        $cont = 0;
        while ($data = pg_fetch_array($result)) {
            foreach ($data as $i => $value) {
                $data[$i] = utf8_decode($value);
            }
            $response[] = $data;
            $cont++;
        }
        $msg = SWATConecta::encerraTransacao();
        if (strcmp($msg, ""))
            return "erro Final Monit::Login";
        if ($cont > 0)
            return $response;
        return "Nome de Usuario ou Senha Invalido";

    }

}

?>
```

Programa 5: Classe Monitor

```
<?php

include_once "SWATConecta.php";

class Professor {

    public static function criaTarefa($codigo_disc,$semestre_disc,
        $ano_disc,$nome,$especpath,$valor,
            $testfilepath,$resultfilepath,$tipo,$supervisionado){

        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "Erro Conexao Professor::cria ";
        }
        $query = "INSERT INTO TAREFA (CODIGO_DISC,SEMESTRE_DISC,ANO_DISC,
            NOME,ESPECIFICACAO,VALOR,TESTE,RESULTADO,TIPO,SUPERVISIONADO)
                VALUES('$codigo_disc',$semestre_disc,$ano_disc,'$nome
                    ','$especpath',$valor,'$testfilepath','
                    $resultfilepath',$tipo,$supervisionado);";

        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::desfazTransacao();
            return "erro Insercao Professor::cria";
        }
        $msg = SWATConecta::encerraTransacao();
        if (strcmp($msg, ""))
            return "erro Final Professor::cria";

    }

    public static function login($RA_Prof,$senha){
```

```
        $msg = SWATConecta::iniciaTransacao();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "Erro Conexao Prof::Login ";
        }
        $query = "SELECT *
        FROM USUARIO
        WHERE RA = $RA_Prof and SENHA = $senha";

        $result = SWATConecta::executa($query);
        $msg = SWATConecta::notice();
        if (strcmp($msg, "")) {
            SWATConecta::encerraTransacao();
            return "erro Login Prof::Login";
        }
        $response;
        $cont = 0;
        while ($data = pg_fetch_array($result)) {
            foreach ($data as $i => $value) {
                $data[$i] = utf8_decode($value);
            }
            $response[] = $data;
            $cont++;
        }
        $msg = SWATConecta::encerraTransacao();
        if (strcmp($msg, ""))
            return "erro Final Prof::Login";
        if ($cont > 0)
            return $response;
        return "Nome de Usuario ou Senha Invalido";

    }

}

?>
```

Programa 6: Classe Professor

```
<?php

include_once 'Aluno.php';
include_once 'Porteiro.php';
include_once 'Professor.php';
include_once 'Secretaria.php';

class SWATBancodeDados {

    public function SWATBancodeDados() {

    }

    public function submete($RA_aluno, $codigo_disc, $semestre_disc,
        $ano_disc, $nome_tarefa, $Lista_filesAdress, $final) {
        return Aluno::submete($RA_aluno, $codigo_disc, $semestre_disc,
            $ano_disc, $nome_tarefa, $Lista_filesAdress, $final);
```

```
    }

    public function historico($RA_Aluno) {
        return Aluno::historico($RA_Aluno);
    }

    public function loginAluno($RA_Aluno, $senha) {
        return Aluno::login($RA_Aluno, $senha);
    }

    public function loginAdmin($RA_Aluno, $senha) {
        return Admin::login($RA_Admin, $senha);
    }

    public function criaDisc($codigo, $semestre, $ano, $nome, $ch,
       $RA_Prof, $RA_Mon) {
        return Admin::criaDisc($codigo, $semestre, $ano, $nome, $ch,
           $RA_Prof, $RA_Mon);
    }

    public function criaUsu($ra, $senha, $nome, $idade, $sexo, $tipo) {
        return Admin::criaUsu($ra, $senha, $nome, $idade, $sexo, $tipo);
    }

    public function matriculaAluno($codigo, $semestre, $ano, $ra_aluno,
       $turma) {
        return Admin::matriculaAluno($codigo, $semestre, $ano, $ra_aluno,
            $turma);
    }

    public function loginMonit($RA_Monit, $senha) {
        return Monitor::login($RA_Monit, $senha);
    }

    public function buscaSubmissoes() {
        return Monitor::buscaSubmissoes();
    }

    public function loginProf($cpf, $senha) {
        return Professor::login($cpf, $senha);
    }

    public function criaTarefa($codigo_disc, $semestre_disc, $ano_disc,
       $nome, $especpath, $valor, $testfilepath, $resultfilepath, $tipo,
       $supervisionado) {
        return Professor::criaTarefa($codigo_disc, $semestre_disc,
           $ano_disc, $nome, $especpath, $valor, $testfilepath,
           $resultfilepath, $tipo, $supervisionado);
    }

}

?>
```

Programa 7: Classe de Comunicação BD/Interface